Towards Grounded Dialogue Generation in Video Game Environments

Nader Akoury, Ronan Salz, Mohit Iyyer

University of Massachusetts Amherst
nsa@cs.umass.edu, rsalz@umass.edu, miyyer@cs.umass.edu

Abstract

Video games provide rich interactive environments that have proven to be great testbeds for AI research in areas such as grounded language generation and reinforcement learning. In this preliminary work, we show that commercial video games can also be an excellent resource for interactive storytelling. We introduce a dataset extracted from the widely acclaimed computer role-playing game *Disco Elysium: The Final Cut*. With roughly 1.1M words of dialogue spread across a complex graph of possible utterances, the game provides a strong research foundation for interactive storytelling. Furthermore, nodes in the dialogue graph can express conditional logic that permanently alter the game state, and thus affects reachability in the graph. These conditions are encoded in the form of Lua scripts written by the game's designers. To demonstrate the utility of the dataset, we cluster dialogue based on similarity and linearize all possible utterances for the next turn of dialogue into a Lua script containing a mix of natural language and game logic. We then mask out one utterance from the script and use a large language model to generate a plausible alternative. Analyses of these generations demonstrate the difficulty of this task and suggest future avenues for research, which if successful, have the potential to profoundly impact dialogue writing in video games.

Introduction

Interactive storytelling allows its consumers to actively guide a story as it unfolds. Broadly speaking, this type of storytelling takes on many forms, including tabletop role-playing games like Dungeons and Dragons (Callison-Burch et al. 2022), choose your own adventure books (Clark and Smith 2021), interactive fiction (Hausknecht et al. 2020), and narrative-driven video games. In this paper, we focus on the latter medium by collecting a dataset from the highly-acclaimed video game *Disco Elysium: The Final Cut*[1] (Kurvitz et al. 2021) and applying large language models (LLMs) to generate dialogue conditioned on the game state.

As such, our work diverges from previous AI-driven interactive storytelling research. Historical approaches have explored encoding stories as graphs and deployed classical planning algorithms (Riedl and Young 2004) to simultaneously incorporate user actions while staying true to authorial intent (Mateas and Stern 2005). Similarly, while commercial video game environments have become increasingly popular testbeds for grounded language (Suhr et al. 2019) and reinforcement learning (Bellemare et al. 2012; Kempka et al. 2016), to the best of our knowledge no commercial video games have been utilized for storytelling research.

Although *Disco Elysium* is a single testcase of the ideas presented in this paper, this task is important. We envision research using the *Disco Elysium* dataset to help produce tools to suggest dialogue options during game design and in-game systems which dynamically react to player input. Additionally, the proposed approach is widely applicable to narrative-driven video games. Numerous video games structure their dialogue in a manner similar to *Disco Elysium* as it uses Pixel Crushers Dialogue System[2], a common framework for video game dialogue. Thus any successful approaches to improving dialogue in *Disco Elysium* can also be applied to those games. Future work could even endeavor to extract dialogue from more of these games to build larger, more diverse datasets.

Disco Elysium provides an interesting dataset for grounded language researchers because it is a large-scale dialogue-driven game: while players control a character in a virtual environment, the majority of a player's interaction takes the form of dialogue grounded in the current game state. We treat *Disco Elysium* as a testcase for other dialogue-driven video games and use it to extract a dataset of dialogue paired with game state references. The extracted dialogue is encoded as a directed graph, with certain nodes acting as boolean gates defined by Lua scripts conditioned on the current state of the game. Additionally, the dataset includes explicit representations of actors, items, game variables, and individual conversations, each of which may contain annotations from the game designers describing their intent.

Given this rich dataset, we devise an approach to use LLMs trained on a mix of code and natural language (Chen et al. 2021; Fried et al. 2022) to diversify the game dialogue while maintaining the designers' vision. We first cluster sim-

[1]Currently rated the #1 PC video game of all time on Metacritic, see https://www.metacritic.com/browse/games/score/metascore/all/pc

[2]https://www.pixelcrushers.com/dialogue-system/

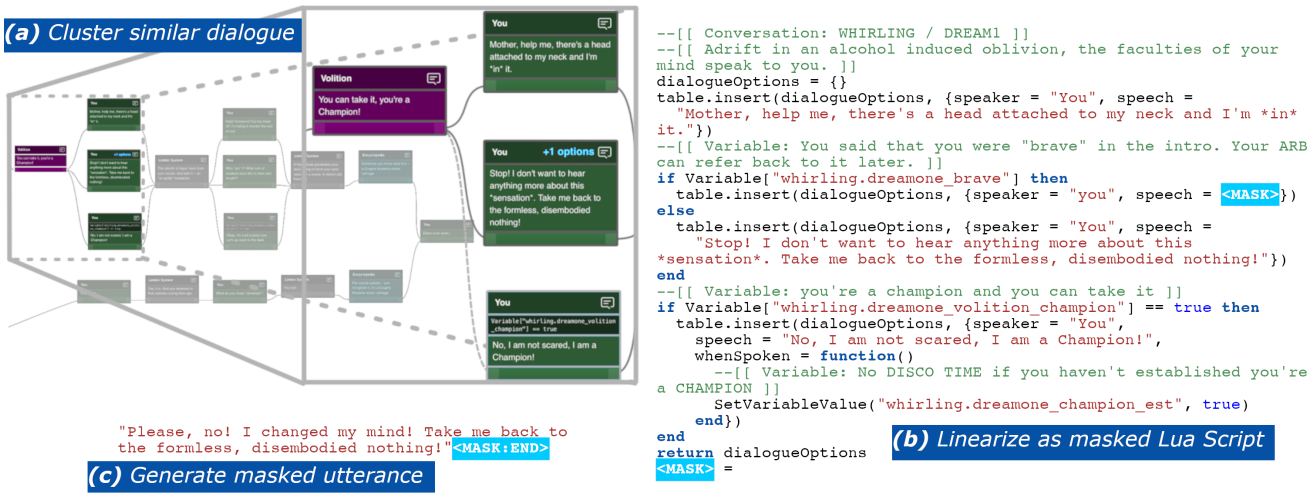

Figure 1: In this example taken from the intro dream sequence of *Disco Elysium: The Final Cut*, we first cluster dialogue nodes by a similarity measure predicated on game state variables and spoken dialogue. Then, we linearize the next turn of dialogue in the graph into a Lua script that contains game logic and dialogue. Finally, we <MASK> one utterance from the cluster and ask a LLM trained on code and natural language to complete the masked dialogue.

ilar dialogue, then linearize the utterances and game state into a Lua script, and ask a LLM to predict a masked line of dialogue (Figure 1). We then conduct a preliminary set of experiments using two LLMs: a finetuned GPT-3 Curie (Brown et al. 2020) model and Codex (Chen et al. 2021) in a few-shot setup. We compare the quality of the generated dialogue using two metrics, BLEURT (Sellam, Das, and Parikh 2020) and a bag-of-words F1. While both metrics indicate Codex performs the task better despite the lack of fine-tuning, the low F1 scores indicate there is a large room for improvement even when only predicting text clustered by similarity.

Dataset

Disco Elysium: The Final Cut is a critically acclaimed dialogue-driven computer roleplaying game where the player takes on the role of a down-on-his-luck detective. There are many genre-defining characteristics of the game that contribute to its popularity, which also make it an excellent resource for interactive storytelling research. The majority of interactions in the game world are in the form of dialogue, which includes interactions with not only other characters but also inanimate objects in the game world (e.g., a ceiling fan and a bathroom mirror from the first scene of the game). Furthermore, the player character has twenty-four attributes that govern in-game skills and also act to convey internal thoughts which frequently interject commentary on the current situation. Finally, the sheer scope of the dialogue (Table 1), combined with the complexity of the choices and their subsequent consequences, make this game an excellent case study to explore interactive storytelling research within an existing commercial video game.

We begin by extracting a catalog of all top-level entities (actors, items, conversations, dialogue entries, and game state variables; see Table 1) from the PC version of *Disco Elysium: The Final Cut* using the open source tool AssetStudio.[3] The game, with over 1.1M words of dialogue across roughly 70K utterances (Table 1), is by most measures of creative fiction one of the longest works in the genre.[4]

Dialogue entries contain references to the actor who is speaking and any precondition, in the form of a boolean-valued Lua expression, required to speak the utterance (see the if statements in Figure 1b), e.g.,

```
Variable["whirling.dreamone_brave"]
```

In addition, each dialogue entry may contain Lua statements which alter game state (see Figure 1b), e.g.,

```
SetVariableValue(
  "whirling.dreamone_champion_est",
  true
)
```

Each top-level entity in the catalog may also contain annotations used internally by the game's designers as a way to document their intent (see the Lua comments in Figure 1b), likely as a way to keep aspects of the complicated story clear during the design process, e.g.,

```
No DISCO TIME if you haven't
established you're a CHAMPION
```

These annotations provide a rich, natural source of context to better explain the vast array of top-level entities (Table 1) used throughout the game logic.

We note that one of the key challenges in using this dataset for research is that *Disco Elysium* represents a single overarching story. For this reason we take special care in splitting

[3] https://github.com/Perfare/AssetStudio

[4] https://gamicus.fandom.com/wiki/List_of_longest_video_game_scripts#Video_games

Dataset Splits

	Train	Valid	Test	Total
	89.8%	5.4%	4.7%	
Utterances	65316	4143	3237	72696
Words	1001191	59877	52816	1113884
Nodes	98442	6950	5092	110484
Forks	17283	1544	964	19791
Variables	99015	6989	5132	111136

(a)

Variable Overlap		Dataset Totals	
Train $\bigcap$ Valid	2897	Actors	424
Train $\bigcap$ Test	2871	Items	259
Valid $\bigcap$ Test	303	Conversations	610

(b) (c)

Table 1: For the *Disco Elysium* dataset: (a) we split the data into training, validation, and test sets based on the number of dialogue words, while ensuring an approximately similar proportion of conditional dialogue forks and referenced Lua variables; (b) we take special care to minimize the number of referenced variable overlaps amongst the splits; (c) though we do not attempt to disentangle Actors and Items across splits.

the dataset into train, valid, and test splits. Considering the high degree of interconnectedness across conversations in the game, it is not possible to create a dialogue split with a disjoint set of actors, items, and game state variables.

As a single variable can be referenced in multiple conversations, the game's dialogue graph implicitly forms a hypergraph, with hyperedges defined by Lua variables. Optimal partitioning of a hypergraph is known to be NP-hard (Papa and Markov 2007), and in smaller hypergraphs an exhaustive enumeration can often be more efficient in practice than specialized algorithms (Papa and Markov 2007). We attempt to generate a 90%/5%/5% train/valid/test split of the conversations in the dataset, while allowing a total $\epsilon = 1.5\%$ variation from the desired splits, using branch and bound to enumerate all valid partitions of the conversations which satisfy the percentage constraints. Luckily, many distinct conversations are connected by one or more dialogue edges, making it such that a handful of connected components in the graph make up roughly 70% of the dialogue (and thus are required to be in the training set). For the remaining dialogue, we opt to minimize the overlap in game variables across splits, as they are crucial for representing conditional dialogue. A final split of 89.8%/5.4%/4.7% is achieved with minimal overlap in variables (Table 1).

Evaluation

To probe the current capabilities of current LLMs to succeed at the proposed task, we first cluster the dialogue nodes to discover utterances which are similar (Figure 1a), but may vary slightly based on the game state. Then, starting from a given dialogue node, we linearize all dialogue nodes potentially reachable in the next turn of conversation to form a Lua script representing all possible utterances and game state altering commands (Figure 1b). Finally, we mask one utterance from the dialogue cluster and have the LLM (GPT-3 Curie or Codex) generate a plausible alternative (Figure 1c).

Clustering

LLMs are often poor at generating relevant continuations for creative text (Akoury et al. 2020). For that reason, we design our preliminary experiments by clustering nodes in the graph by similar text that only varies slightly based on game state (see Figure 1), e.g.,

(a) `"Stop! I don't want to hear anything more about this *sensation*. Take me back to the formless, disembodied nothing!"`

(b) `"Please, no! I changed my mind! Take me back to the formless, disembodied nothing!"`

We experiment with a number of algorithms for clustering the game's dialogue, including the Levenshtein distance, Jaccard index, and the Dice coefficient (equivalent to a bag-of-words F1). We also vary the features used for clustering by splitting the words into characters, grouping by ngrams, and through the use of lowercasing. We conduct a manual inspection of the various approaches to clustering, including a hyperparameter sweep of the similarity threshold. This inspection indicated that solely clustering based on the dialogue utterances would either systematically miss semantically similar text, or cluster dissimilar utterances when the similarity threshold was made more permissive.

To combat this tendency, we additionally tried clustering nodes by inspecting the associated Lua conditions. We first parse the Lua expression, extract identifiers (which often refer to functions) and string literals (which often refer to variables). We then split the literals into their constituent words (e.g., `whirling.dreamone_brave` becomes `whirling, dreamone, brave`), before running the above battery of clustering approaches. In the end, we find that clustering based on a combination of dialogue and Lua expressions produced the best results, without the need for the extra feature engineering, while only relying on the simple Dice coefficient with a threshold $d >= 0.5$.

Model Class	Prompt Tokens	Model Type	OpenAI API Name
Curie [5]	2048	Finetuned	`curie`
Codex	8000	Few-Shot	`code-davinci-002`

Table 2: Details of the models used in our experiments. As OpenAI does not provide parameter counts or details on finetuning, we also provide the API name for the models to help reproducibility.

[5] Likely 6.7B parameters, see: https://blog.eleuther.ai/gpt3-model-sizes/

Model	Examples	Tokens	BLEURT	F1
Curie	2,668	3,041,299	41.9	25.6
Codex	2,668	21,077,200	44.2	29.5

Table 3: Preliminary experiments over the validation set show that few-shot Codex outperforms a finetuned Curie model for generating context-aware dialogue.

Model	Examples	Copied
Curie	2,668	235
Codex	2,668	455 (8)†

Table 4: We find that both Curie and Codex occassionally copy dialogue from the prompt, and in $8^{\dagger}$ instances Codex directly copies a completion from the few-shot examples.

Linearization and Masking

Now that we have a set of clusters, we need to convert each cluster into a Lua script that can be fed to a language model. We do that by prefixing all the top-level entities (e.g. actors, conversations, and variables) referenced by the clustered nodes at the top of the script, and any default value they may have. We additionally include any annotations by the game's designers in the form of Lua comments. Each node in the cluster is visited in sequential order and its Lua conditions, dialogue, and any associated actions when the dialogue is spoken is included in the script. Lastly, we generate all variants of each script by enumerating every clustered utterance from the prompt, masking them out one-by-one, and asking the LLM to generate the masked dialogue as its completion (see Figure 1bc). We use a prefix-suffix-mask (Donahue, Lee, and Liang 2020) ordering of the masked infilling examples rather than a suffix-prefix-mask (SPM) form, as SPM only achieves better performance when pretraining on billions of tokens (Bavarian, Jun, and Tezak 2022).

Experiments

We conduct experiments using two LLMs: GPT-3 Curie and Codex (Table 2). GPT-3 Curie is a strong generation model for natural language (Brown et al. 2020), especially when finetuned on a downstream task, while Codex is an extremely capable few-shot LM for code (Chen et al. 2021). As our task contains elements of both natural language and code, it is important to assess the capabilities of each model paradigm.

Since the two models perform different tokenization[6] and support different context lengths, we filter the clusters, keeping only those that fit the smallest context length (2048 tokens) using the GPT-3 tokenizer. We then generate all the linearized scripts representing semantically related text for the next turn of dialogue. After filtering and generating masked variants of the clusters, we are left with 30,501 training examples and 2,668 validation examples.

[6]Codex uses a modified tokenizer that collapses whitespace since it is commonly used in code formatting.

We finetune Curie for 1 epoch, with a batch size of 32 examples and a learning rate of $0.2\times$ the learning rate of the pretrained model and we weight the loss for the prompt tokens by 0.01. For the few-shot Codex model, we prefix each linearized Lua script with several samples from the validation set such that they take up nearly the full context window (we reserve 100 tokens of the context for generation). We also ensure there are no overlaps in dialogue between the few-shot examples and the script. Consequently, each Codex script has 7 few-shot examples on average.

We choose to measure the performance of the models on the validation set using a bag-of-words F1, as the clustered utterances have a large overlap with the masked text the model is tasked with infilling. In addition, we use BLEURT which has proven to be robust for semantic similarity of generated text (Karpinska et al. 2022). Both metrics favor Codex slightly, though given the low F1 score, it's clear the models have much room for improvement on this simplified form of our task. That is to say, naïvely applying our preliminary approach to all the dialogue in the game, not just to the subset of dialogue clustered via similarity, is even more likely to fail. We also posit that Codex likely outperforms Curie since it is a larger model that is explicitly trained on a large corpus of code, even though it uses a few-shot approach to inference.

Analysis

To better understand the performance difference between the two models we also conduct a small analysis of each model's output. We find that both models tend to copy from the prompt (Table 4), but Codex does it nearly twice as often.

A qualitiative inspection of the generations from the Codex model (our best performer) seem to indicate the model may struggle to generate plausible completions due to a lack of historical context to the current conversation. Our script-based prompts do not include any previous dialogue utterances, but rather rely only on the combination of dialogue that can be emitted next and conditional game logic gating those options. It is clear the models also do not make effective use of the game designer's annotations to fill in the gaps. While these comments are likely useful reference for the writers of the game, they may not contain enough context alone to guide generation. Considering Codex has a very long context window and performs better than a finetuned Curie (Table 3), future experiments could attempt to include previous turns of dialogue in the prompt to see if that improves generation quality.

Future Directions

While these preliminary experiments provide useful insights, automatic metrics of creative writing are known to poorly correlate with human judgements (Karpinska, Akoury, and Iyyer 2021). For that reason, we have already developed a webapp with an embedded Lua VM which recreates the in-game dialogue system while incorporating generated dialogue from LLMs. Having a test harness separate from the game environment allows for a controlled human evaluation protocol. We plan an interactive user-study along with further modeling improvements.

References

Akoury, N.; Wang, S.; Whiting, J.; Hood, S.; Peng, N.; and Iyyer, M. 2020. STORIUM: A Dataset and Evaluation Platform for Machine-in-the-Loop Story Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6470–6484. Online: Association for Computational Linguistics.

Bavarian, M.; Jun, H.; and Tezak, N. 2022. Efficient Training of Language Models to Fill in the Middle. *arXiv:2207.14255 [cs]*, 30.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2012. The Arcade Learning Environment: An Evaluation Platform for General Agents. *CoRR*, abs/1207.4708.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models Are Few-Shot Learners. *arXiv:2005.14165 [cs]*.

Callison-Burch, C.; Tomar, G. S.; Martin, L. J.; Ippolito, D.; Bailis, S.; and Reitter, D. 2022. Dungeons and Dragons as a Dialog Challenge for Artificial Intelligence. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 15.

Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; Pinto, H. P. d. O.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; Ray, A.; Puri, R.; Krueger, G.; Petrov, M.; Khlaaf, H.; Sastry, G.; Mishkin, P.; Chan, B.; Gray, S.; Ryder, N.; Pavlov, M.; Power, A.; Kaiser, L.; Bavarian, M.; Winter, C.; Tillet, P.; Such, F. P.; Cummings, D.; Plappert, M.; Chantzis, F.; Barnes, E.; Herbert-Voss, A.; Guss, W. H.; Nichol, A.; Paino, A.; Tezak, N.; Tang, J.; Babuschkin, I.; Balaji, S.; Jain, S.; Saunders, W.; Hesse, C.; Carr, A. N.; Leike, J.; Achiam, J.; Misra, V.; Morikawa, E.; Radford, A.; Knight, M.; Brundage, M.; Murati, M.; Mayer, K.; Welinder, P.; McGrew, B.; Amodei, D.; McCandlish, S.; Sutskever, I.; and Zaremba, W. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374.

Clark, E.; and Smith, N. A. 2021. Choose Your Own Adventure: Paired Suggestions in Collaborative Writing for Evaluating Story Generation Models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 3566–3575. Online: Association for Computational Linguistics.

Donahue, C.; Lee, M.; and Liang, P. 2020. Enabling Language Models to Fill in the Blanks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2492–2501. Online: Association for Computational Linguistics.

Fried, D.; Aghajanyan, A.; Lin, J.; Wang, S.; Wallace, E.; Shi, F.; Zhong, R.; Yih, W.-t.; Zettlemoyer, L.; and Lewis, M. 2022. InCoder: A Generative Model for Code Infilling and Synthesis. arXiv:2204.05999.

Hausknecht, M.; Ammanabrolu, P.; Côté, M.-A.; and Yuan, X. 2020. Interactive Fiction Games: A Colossal Adventure. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05): 7903–7910.

Karpinska, M.; Akoury, N.; and Iyyer, M. 2021. The Perils of Using Mechanical Turk to Evaluate Open-Ended Text Generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 1265–1285. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Karpinska, M.; Raj, N.; Thai, K.; Song, Y.; Gupta, A.; and Iyyer, M. 2022. DEMETR: Diagnosing Evaluation Metrics for Translation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Kempka, M.; Wydmuch, M.; Runc, G.; Toczek, J.; and Jaśkowski, W. 2016. ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning. *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8.

Kurvitz, R.; Hindepere, H.; Tuulik, A.; De Cuir, C.; and Moskvina, O. 2021. *Disco Elysium: The Final Cut*. ZA/UM Studios.

Mateas, M.; and Stern, A. 2005. Demonstration: The Interactive Drama Façade. In *Proceedings of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE'05, 153–155. Marina del Rey, California: AAAI Press.

Papa, D. A.; and Markov, I. L. 2007. Hypergraph Partitioning and Clustering. In *Handbook of Approximation Algorithms and Metaheuristics*.

Riedl, M. O.; and Young, R. M. 2004. An Intent-Driven Planner for Multi-Agent Story Generation. *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004.*, 186–193.

Sellam, T.; Das, D.; and Parikh, A. 2020. BLEURT: Learning Robust Metrics for Text Generation. In *Theoretical Issues in Natural Language Processing-2*, 7881–7892. Online: Association for Computational Linguistics.

Suhr, A.; Yan, C.; Schluger, J.; Yu, S.; Khader, H.; Mouallem, M.; Zhang, I.; and Artzi, Y. 2019. Executing Instructions in Situated Collaborative Interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2119–2130. Hong Kong, China: Association for Computational Linguistics.